평면 형상에 대한 꼬임의 발생이 배제된 계층적 표현

○
허봉식*, 김성옥**, 김민환***
*동의공업대학 영상정보과
**한국신발피혁연구소
***부산대학교 컴퓨터공학과

A Hierarchical Representation Scheme without Self-Intersection for Planar Shapes

○
Pong-Sik Ho*, Seong-Ok Kim**, Min-Hwan Kim***
*Dept. Visual Technologies, Dongeui Institute of Technology
**Korea Institute of Footwear and Leather Technology
***Dept. of Computer Engineering, Pusan National University
E-mail: psho@dit.ac.kr*, sokim@kiflt.re.kr**, mhkim@hyowon.pusan.ac.kr***

요 약

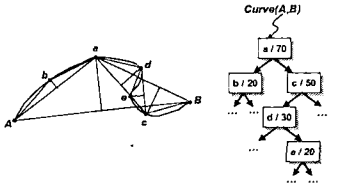
본 논문에서는 평면 형상에 대한 꼬임(self-intersection)의 발생을 배제한 계층적 표현을 제안하였다. 기존의 계층적 표현들에서는 실제 활용에 많은 문제를 초래하는 꼬인 형상의 발생을 피할 수 없으며, 이는 반드시 제거될 필요가 있다. 그러나 n개의 점으로 구성된 형상의 꼬임 문제를 해결하는 데는 $O(n^2)$의 계산량이 요구되므로, 본 논문에서는 계층적 지역화의 경계 영역간 교차 상태를 나타내는 cross-link 개념을 도입하여 효율적으로 해결할 수 있는 꼬임 해결 알고리즘을 제안하고 실험을 통해 그 효율성을 확인하였다.

1. 서론

컴퓨터 비전, 패턴 인식, 지도 제작, 컴퓨터 그래픽 등의 응용 분야들에서는 객체 형상에 대한 계층적 표현이 매우 유용하게 사용될 수 있다. 객체 형상에 대한 계층적 표현은 다양한 정밀도의 표현을 효율적이면서도 체계적으로 트리 구조를 이용하여 나타내며, 필요에 따라 임의의 정밀도의 표현을 생성해서 사용할 수 있도록 하기 때문이다. 이것은 형상에 대한 중요하지 않은 상세한 부분에 대한 처리를 응용 상황에 따라 생략할 수 있게 함으로써 형상 처리 속도를 높이는데 중요한 요소로 작용한다.

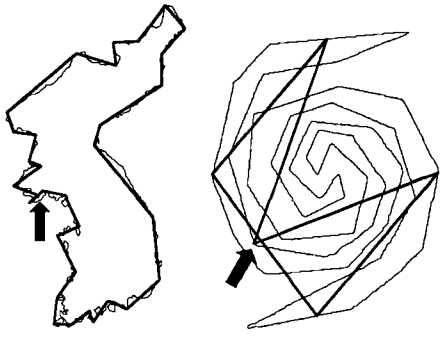
계층적 표현 관련 연구로는 Ballard의 strip tree[1], Günther의 arc tree[2,3], Veltkamp의 HAL tree[4] 등이 있다. 그러나, 이 방법들은 계층적 표현들이 지연스럽지 못하거나, 표현 정밀도의 선택 척도가 적절하지 않고, 형상에 대한 기하학적 연산의 효율을 저하시키는 경계 영역간의 포함 문제 등을 내포하고 있다[5].

그리고 [5]에서 이러한 문제들을 극복한 새로운 계층적 표현 방법을 제시하였다. 제시된 방법은 Douglas-Peucker 알고리즘을 기반으로 하여 자연스러운 형상 표현을 할 수 있도록 계층적 근사화를 하였고 또한, MBO(Minimum Bounding Octangle)을 경계 영역으로 하여 효율적인 계층적 지역화 성능을 가진 새로운 계층적 표현이다. 계층적 근사화 과정에서는 곡선 분할 점 계산 과정의 근사화 오차를 분할 점의 속성 값으로 트리 노드에 같이 저장하여 주어진 오차 임계 값을 가진 근사화 표현을 생성할 때 사용할 수 있도록 하였다.



[그림 1] 계층적 근사화 과정

그러나 관련 연구에서 해결되지 않은 중요한 문제는 계층적 표현으로부터 생성된 폴리곤에 그림 2과 같이 꼬임(self-intersection)이 발생할 수 있다는 것이다. 아래의 그림에서 확인할 수 있듯이, 꼬임은 복잡한 형상에서 뿐만 아니라 비교적 상세한 작은 부분에서도 발생할 수 있다는 것을 알 수 있다. 이렇게 꼬임이 포함된 표현은 형상에 대한 면적, 둘레 길이, 내외부 판별 등과 같은 기하학적 연산에 사용할 수 없으므로 실질적인 활용 가치가 없다고 볼 수 있다.



[그림 2] 꼬임이 포함된 폴리곤 생성 예

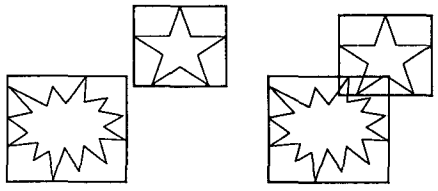
이에 본 논문에서는 계층적 표현의 가능한 모든 표현에서 꼬임의 발견과 해결을 효율적으로 처리할 수 있는 꼬임 해결 알고리즘(self-intersection resolving algorithm)을 제안하고 실험을 통해 그 유용성을 확인하였다. 2장에서는 제안한 알고리즘을 상세히 설명하였고, 3장에서는 실험 결과와 이에 대한 분석을 하고 마지막으로 결론을 맺는다.

2. 제안한 꼬임 해결 알고리즘

2.1 Cross-link 개념

주어진 형상에 대한 계층적 표현에서 꼬임의 발생을 미리 예측하는 것은 불가능하며 모든 가능한 표현에 대해 꼬임 발생 여부를 검사해야 한다. 그러므로 n개의 점으로 구성된 폴리곤의 계층적 표현에 대한 꼬임 발생 검사는 $O(n^2)$의 계산량이 일반적으로 요구된다. 이것은 실제 활용에 부담이 되며 보다 효율적인 검사 방법이 필요하다.

이에 본 논문에서는 [5]에서 사용한 계층적 지역화를 기반으로 cross-link 개념을 도입하였다. 주어진 객체의 경계 영역은 대상 객체의 위치와 크기에 관한 정보를 단순화 시킨 도형으로 표현한다. 그러므로 복잡한 두 도형간의 교차 여부는 이들에 대한 경계 영역들의 교차 여부를 먼저 검사함으로써 교차하지 않는 경우를 쉽게 판단할 수 있다(그림 3(a)). 그리고 경계 영역들 간의 교차는 형상들간의 교차 가능성을 의미하므로 보다 상세한 검사를 해야 실제 교차 여부를 알 수 있다(그림 3(b)).

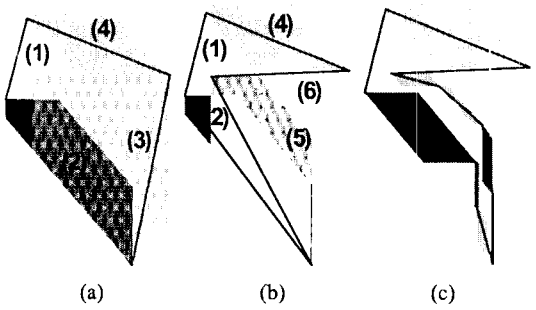


[그림 3] (a) 경계 영역간 교차가 없는 경우, (b) 경계 영역 간의 교차가 발생한 경우

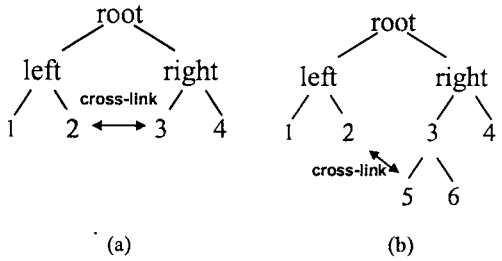
본 논문에서 제안하는 cross-link는 교차 상태에 있는 두 경계 영역(MBO)을 말한다. 위에서 언급한

바와 같이, cross-link된 두 경계 영역은 실제 형상들간의 교차 가능성 있다는 것을 의미하며, 최종 교차 여부를 확인하기 위해서는 한 단계씩 상세한 검사를 해야 함을 의미한다. 또한 cross-link되지 않은 MBO에 포함된 형상들간에는 교차가 없을 뿐만 아니라, 이들에 대한 가능한 모든 근사화 표현들에 대해서도 서로 교차할 가능성은 없다고 단정할 수 있다. 이와 같이 cross-link를 이용하면 초기에 교차 가능성이 없는 부분들을 검사대상에서 제외할 수 있게 됨으로써 처리속도를 개선할 수 있다.

그림 4은 MBO를 이용한 계층적 경계 영역들의 확장 과정을 나타낸다. 그림에서의 각 MBO는 현재의 근사화 표현뿐만 아니라, 이로부터 가장 상세하게 표현될 수 있는 부분까지의 전체 형상을 포함한다. 그러므로 상위 노드의 MBO들간의 교차가 없다면 두 MBO의 모든 하위 노드들의 MBO간에도 교차가 없다. 또한, 관련된 근사화 표현까지도 교차가 없다는 것을 의미한다. 그림 4(a)에서 MBO 2와 MBO 3이 서로 교차하여 cross-link가 존재함을 그림 5(a)에 트리 구조로 표현하였다. 일단 발생한 cross-link는 관련된 두 노드들 중에서 오차가 큰 노드를 두 개의 자식 노드로 확장하면서 제거되며, 확장된 두 노드와 이전의 나머지 노드의 경계 영역들 간에 교차 여부를 확인하여 새로운 cross-link를 만들게 된다. 또한 새로 확장된 두 노드에 대해서도 같은 검사를 한다. 그림 4(b)와 그림 5(b)는 노드 3이 노드 5와 노드 6으로 확장된 후에 노드 2와 노드 5 사이에 새로운 cross-link가 발생하였음을 보여준다. 이와 같이 cross-link가 없어질 때까지 경계 영역을 확장하면 그림 4(c)와 같은 상태가 된다. 이 경우에는 모든 경계 영역들간에는 서로 교차가 없으므로 각 경계 영역 내부에서의 꼬임 발생 가능성만 확인하면 된다. 그러므로 복잡한 문제가 보다 단순한 여러 개의 문제들로 분할되어 결과적으로 전체적인 처리 성능의 향상을 가져오게 된다.



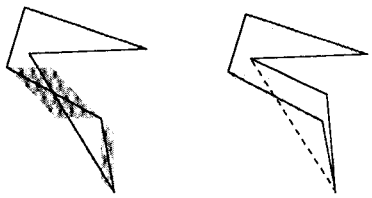
[그림 4] 계층적 경계 영역의 확장 과정



[그림 5] Cross-link의 발생과 갱신 과정

2.2 꼬임의 해결

Cross-link를 이용하여 꼬임 가능성이 있는 부분들만을 추적하다가 실제 근사화 표현에서 꼬임(self-intersection)이 발생한 경우에는(그림 6(a)) 꼬인 노드들 중에서 큰 오차를 가진 노드를 확장하여 보다 상세하게 표현한다(그림 6(b)). 이 과정에서 확장된 두 자식 노드의 오차는 부모의 오차를 물려받아 부모 노드가 형상 표현에 포함되는 경우에는 항상 같이 포함되도록 하여 꼬임이 나타나지 않게 된다. 물론 한번의 확장으로 꼬임이 제거되는 경우도 있지만, 만일 확장 결과, 추가의 꼬임이 새로 발생하는 경우에는 꼬임이 제거될 때까지 꼬인 노드들을 재귀적으로 확장하여야 한다. 이 과정에서 확장된 모든 노드들에 대한 오차는 최초의 꼬임 해결을 위해 사용된 오차로 갱신한다.



(a) (b)

[그림 6] 꼬임 해결을 위한 경계 영역의 확장 예

2.3 꼬임 해결 알고리즘

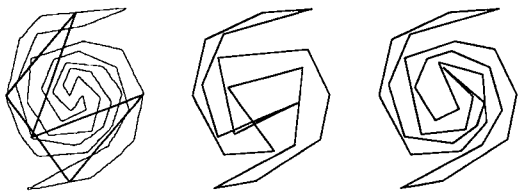
지금까지의 꼬임 해결 과정에 대한 상세한 알고리즘을 정리하면 다음과 같다. 주어진 형상에 대한 계층적 표현의 루트 노드를 *H*라 하면, 이에 대한 꼬임 제거 알고리즘은 *HRESOLVE(H)*로 시작된다.

```
ALGORITHM HRESOLVE (Node H)
  Input: Hierarchical tree, H.
  Output: The same tree without self-intersections.
  Initialize variables to Null or zero;
  Add H to NOT_CROSS_LINKED_SET;
  While NOT_CROSS_LINKED_SET is not empty {
     N ← Get a node from NOT_CROSS_LINKED_SET;
     EXPAND_CROSS_LINK (N);
     EXPAND_TREE ();
  }

ALGORITHM EXPAND_TREE ()
  N ← A node with maximum attribute from
       CROSS_LINK_SET or NULL if the set is empty;
  While N is not NULL {
     EXPAND_CROSS_LINK (N);
     If SELF_INTERSECTIONS > 0 {
        Set RESOLVING_ATT with the attribute of N;
        RESOLVE_TREE ();
     }
  }

ALGORITHM RESOLVE_TREE ()
  While SELF_INTERSECTIONS > 0 {
     N ← Select a node with maximum attribute from
          CROSS_LINK_SET;
     SELF_INTERSECTIONS --;
     EXPAND_CROSS_LINK (N);
     Add nodes N, N.left and N.right to ATT_ADJ_LIST;
  }
  Update the attributes of all nodes of ATT_ADJ_LIST with
       RESOLVING_ATT;

ALGORITHM EXPAND_CROSS_LINK (Node N)
  CHECK_ADD_CROSS_LINK (N.left, N.right);
  CLN_LIST ← List of nodes that are cross-linked with N;
  For each M in CLN_LIST {
     CHECK_ADD_CROSS_LINK (M, N.left);
     CHECK_ADD_CROSS_LINK (M, N.right);
  }

ALGORITHM CHECK_ADD_CROSS_LINK (Node L, R)
  If the MBOs of L and R are intersected {
    If approximated lines of L and R are intersected {
      Add a new intersected cross-link, (L, R,
          INTERSECTED), to CROSS_LINK_SET;
      SELF_INTERSECTIONS ++; }
    Else {
      Add a new cross-link, (L, R, NULL), to
          CROSS_LINK_SET; }
  }
  Else
    Add L and R to NOT_CROSS_LINKED_SET;
```

3. 실험 및 분석

3.1 굴곡이 많은 형상을 통한 꼬임 해결 예

그림 7(a)는 인위적으로 복잡하게 만든 형상의 계층적 표현에서 첫번 째 꼬임이 발생한 표현이다. 이를 해결하기 위해 꼬인 노드를 확장하면 새로운 꼬임이 발생하며, 그 교차의 개수는 확장할수록 오히려 증가하게 된다. 그림 7(b)는 15번 확장한 후에도 아직 꼬임이 남아 있음을 보여주며, 29번을 확장한 뒤(그림 7(c))에 모든 꼬임이 해결되었다. 그리고 이 과정에 나올 수 있는 모든 꼬인 표현들은 꼬임 해결 알고리즘 적용 과정에서 모두 제거되어(오차 상속을 통해서) 다음부터는 생성되지 않는다.



(a) (b) (c)

[그림 7] (a) 꼬임 발생, (b) 15번 확장, (c) 29번 확장

3.2 가능한 표현 수의 비교

꼬임 해결 알고리즘은 기본적으로 계층적 트리에 저장된 오차의 조정 과정이라고 볼 수 있다. 꼬임 해결 과정에서 부모 노드의 오차 값을 상속 받게 되어 있으므로, 계층적 표현으로부터 근사화 폴리곤 생성에 사용할 수 있는 오차 임계 값의 선택 폭이

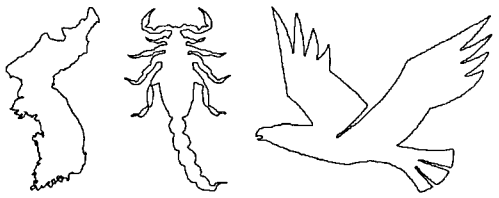
줄어들게 된다. 그러나 이로 인해 꼬임이 완전히 배제된 형상을 얻을 수 있으므로 계층적 표현의 유용성은 더욱 커진다고 볼 수 있다.

표 1은 그림 8의 5개 형상을 대상으로 실험한 결과이다. Douglas-Peucker 알고리즘을 토대로 한 계층적 표현(DP algorithm), 자연스러운 표현을 위해 오차 조정 알고리즘을 적용한 경우(Natural representation)[5], 꼬임까지 모두 제거한 경우(Self-intersection resolved) 등 세가지 경우를 비교하였다. 우선, 자연스러운 표현을 위한 오차 조정 과정에서 표현 가능한 가지 수가 줄어 들었고, 꼬임 해결을 위한 오차 조정 과정에서 또한 많이 줄어든 것을 볼 수 있다.

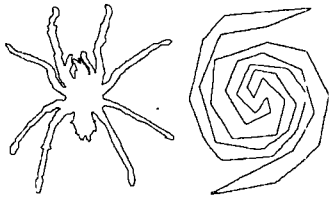
[표 1] 계층적 표현으로부터 생성 가능한 표현 수

Image Name	DP algorithm	Natural representation	**Self-Intersection resolved.**
Korean map	319	279	**272**
scorpion	297	265	**237**
bird	122	106	**94**
spider	246	206	**202**
spiral	153	105	**62**

생성 가능한 표현 수가 줄어드는 정도는 형상 자체의 굴곡이 많을수록 (예: spiral) 커지는 것을 알 수 있고, spider나 Korean map의 경우에는 꼬임 해결 과정에서 거의 줄어들지 않았음을 알 수 있다. 그 이유는 Korean map은 굴곡이 심하지 않은 비교적 단순한 형상이고, spider는 복잡해 보이기는 하지만 방사형 구조로 되어 있어 꼬임이 발생할 확률이 상대적으로 낮기 때문으로 분석된다.



(a) (b) (c)



(d) (e)

[그림 8] (a) Korean map, (b) scorpion, (c) bird, (d) spider, (e) spiral

4. 결론

본 논문에서는 형상에 대한 계층적 표현에서 꼬임을 효율적으로 제거할 수 있는 꼬임 해결 알고리즘을 제안하였다. 계층적 지역화의 경계 영역으로 사용된 MBO간의 교차 상태를 cross-link 개념을 도입하여 효율적으로 관리하면서 실제 꼬임 발생 가능성이 있는 부분들에 대해서만 계산할 수 있는 알고리즘을 구현하였으며 실험을 통해 그 유용성을 확인하였다.

[참고문헌]

[1] D.H. Ballard, "Strip trees: A Hierarchical Representation for Curves", Communications of ACM, Vol.24, No.5, pp.310-321, 1998

[2] O. Günther and E. Wong, "The Arc Tree: An Approximation Scheme to Represent Arbitrary Curved Shapes", Computer Vision, Graphics, and Image Processing, Vol. 51, pp. 313-337, 1990

[3] O. Günther and S. Sominguez, "Hierarchical Schemes for Curve Representation", IEEE Computer Graphics and Application, Vol. 13, Issue 3, pp.55-63, 1993

[4] R.C. Veltkamp, "Hierarchical approximation and localization", The Visual Computer, Vol. 14, pp. 471-487, Springer Verlag, 1998

[5] 허봉식, 김민환 , "평면 형상에 대한 새로운 계층적 표현 방법", 한국멀티미디어학회 2001년 춘계학술발표논문집, 부산, pp. 212-215, 2001